National University of Computer and Emerging Sciences

Lab Manual

for

Web Engineering (SL3003)

Course Instructor	Mr. Ayaz Gillani
Lab Instructor	Ahmad Jawad Mustasim
Lab Demonstrator	Mohid Jillani
Section	6A
Semester	Spring 2025

Department of Software Engineering

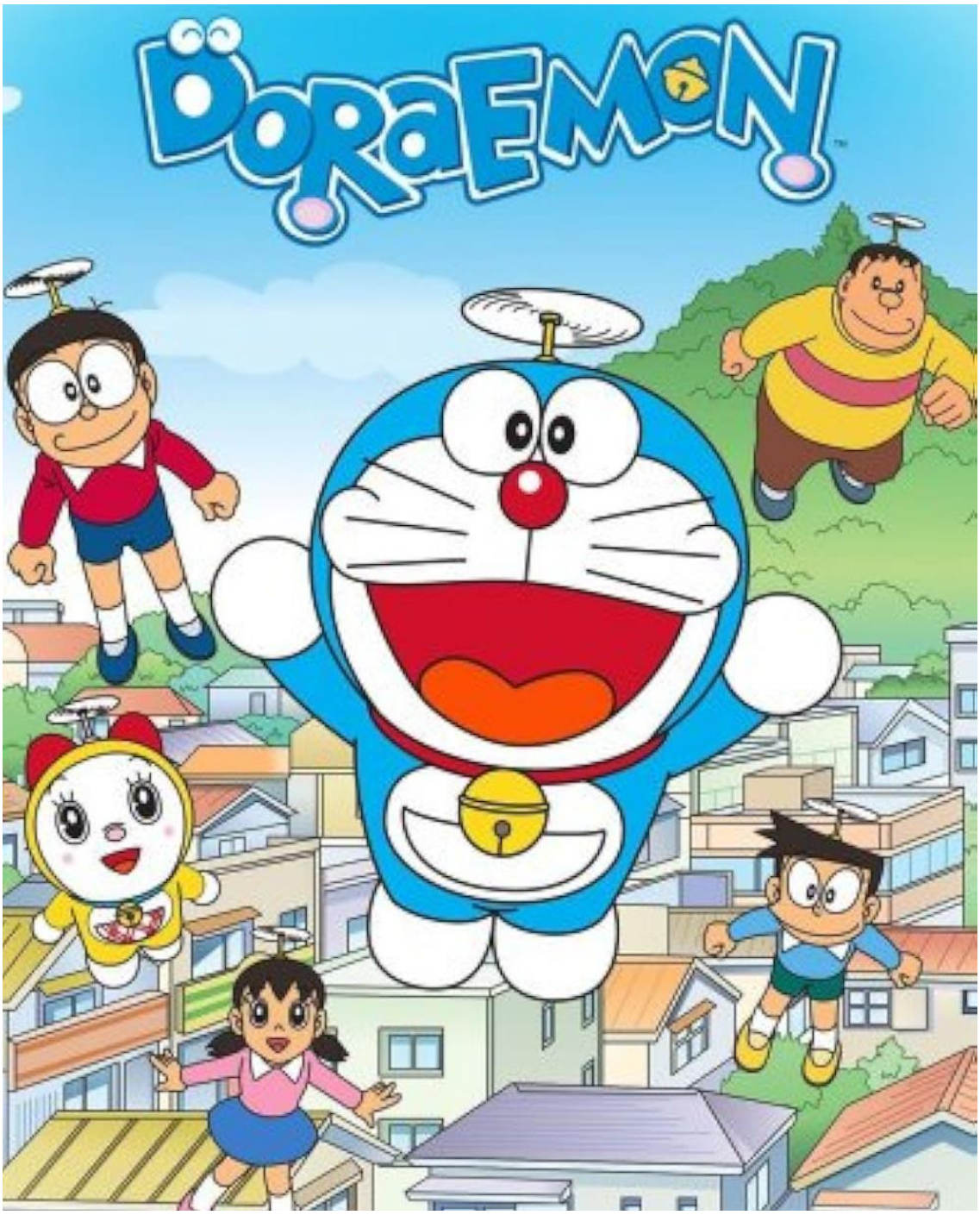
FAST-NU, Lahore, Pakistan

Lab 13 – Express JS APIs

Objectives

- Backend integration with MongoDB
- CRUD Operations (GET, POST, PATCH, PUT, DELETE)

Doraemon Gadget Center



Use following tools/soft wares:

- VS Code
- Postman

Setup your project:

- Firstly, setup node project: **npm i init –y**
- Now install express: **npm install express**
- For automatic restart of server after you save something after updating your code, you can install nodemon: **npm install --save-dev nodemon**

File structure of your project should be:

```
doraemon-gadget-lab/
├── models/
│   ├── Character.js          # object for characters (robot/human)
│   └── Gadget.js             # object for gadgets
│
├── routes/
│   ├── characterRoutes.js    # Character-related endpoints
│   └── gadgetRoutes.js       # Gadget-related endpoints
│
├── controllers/              # (Optional but clean for organizing logic)
│   ├── characterController.js  # Logic for handling characters
│   └── gadgetController.js     # Logic for handling gadgets
│
├── index.js                  # Entry point - sets up server and routes
├── .env                      # MongoDB URI and config variables (if used)
├── package.json              # Project dependencies and scripts
└── README.md                 # Lab instructions or documentation
```

Lab Tasks

Note:

- Test all your APIs on POSTMAN and attach screenshots also.

1. Create a character object with fields: id (auto-incremented), charactername and role (should be enum: "human" or "robot").

2. Create a gadgets object with fields: id (auto-incremented), name, description, use timestamps (for createdAt and updatedAt) and addedBy (character's _id).

3. Implement a route POST /character to register a new character (robot or human).

   Example: Add 2 robots: Dorami and Doraemon
   Add 2 humans: Nobita and Shizuka

4. Implement a route POST /gadgets (the addedBy field should be filled with the character's ID)

   Example: Add Doraemon's bamboo-copter and anywhere-door. (addedBy field should be character_id of Doraemon)

5. Implement a route GET /characters that returns all characters.

6. Implement a route GET /gadgets that returns all gadgets.

7. Implement a route GET /gadgets/:id that returns a single gadget.

8. Implement a route GET /character-gadgets/:id that returns all gadgets by a specific character. (in case of human character return error response that humans don't own gadgets)

9. Implement a route PATCH /gadgets/:id to update a gadget

10. Implement a route DELETE /gadgets/:id to delete a gadget

11. Implement GET /gadgets?name=bamboo to filter gadgets by name using req.query.

12. Use appropriate response types with correct codes

Code	Type	Meaning	When to Use
200	✓✓ Success	OK – Request succeeded	GET, PUT, PATCH, DELETE operations when everything is fine
201	✓✓ Created	Resource created	POST request when a new user or gadget is successfully created
204	✓✓ No Content	Success, no data to return	DELETE request when item is successfully removed
400	✕ Client Error	Bad Request	Invalid input, missing required fields
401	🔐 Unauthorized	Authentication required	Used when user is not logged in
403	⛔ Forbidden	Access denied	e.g. normal user tries to access admin functionalities
404	? Not Found	Resource not found	e.g. user ID doesn't exist
500	✹ Server Error	Internal server error	Uncaught error, DB failure, server crash